Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 13 (2011) 531–537

Engineering Procedia

Conference Title

Analysis of the Lawn Bowl Trajectory as a teaching tool for Sports Engineering: development of a graphical user-interface

Paul R. Medwell[a]*, Laura A. Brooks[a] and Barry S. Medwell[b]

[a] *School of Mechanical Engineering, The University of Adelaide, S.A. 5005, AUSTRALIA*

[b] *Kingston Community School, Kingston SE, S.A. 5275, AUSTRALIA*

Abstract

Sports engineering graduates require a range of competencies. The ability to interactively process and analyse data is a useful skill that may not be fully covered in more traditional engineering subjects. One particular focus within this area that many graduates will find invaluable is the ability to write computer programmes incorporating a graphical user-interface (GUI). This paper outlines the development of a GUI in MATLAB® to interactively present the trajectory of a lawn bowl. Through the developed MATLAB® GUI, numerous parameters can be varied and the influence on the trajectory of a lawn bowl can be shown graphically. This practical example also provides an excellent opportunity to teach many other relevant aspects of MATLAB®. The skills that are introduced in this paper may readily be applied to a range of other sports engineering applications.

Keywords: Sports Engineering Education ; Lawn Bowls ; Graphical User-Interface

Nomenclature

d	Distance between centre of mass and geometric centre	r_0	Initial radius of curvature	v_x, v_y	Velocity of bowl in Cartesian coordinates
g	Gravitational acceleration	R	Radius of bowl	v_o	Launch velocity of bowl
I_0	Moment of inertia	t	Time relative to launch of ball	x, y	Cartesian coordinates
M	Mass of bowl	T	Time until ball comes to rest	ϕ	Angle swept by ball
P	Non-dimensional grouping of constants	v	Velocity of bowl	μ	Coefficient of rolling friction

* Corresponding author. Tel.: +61 (0)8 8303 5460; fax:+61 (0)8 8303 4367.
E-mail address: paul.medwell@adelaide.edu.au

1877–7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.05.126

1. Introduction

With the emergence of sports engineering as a recognised engineering Bachelor degree, with the first example in Australia being the University of Adelaide, there is a growing need for suitable teaching material. It is well-known that developing core fundamental engineering skills in the areas of mathematics and dynamics is important for sports engineering graduates [1]. Sport provides many real-world scenarios that can be used to teach a range of fundamental skills in the broad areas of mathematics and physics [2]. There are countless examples that are well suited to application of the fundamental principles of the equations of motion in a sporting context [3]. One specific example is the sport of lawn balls, as presented by Cross [4].

Although sports engineering programmes typically incorporate aspects of existing courses in the broad fields of mechanical and electronic engineering, there are other skills that are specifically required for sports engineers to possess. As an example, since sports engineers deal directly with athletes and coaches, it is essential that data can be inputted and displayed on a computer quickly, efficiently and interactively. To meet this need, sports engineers should be competent with the development of graphical user-interfaces (GUIs). With many sports engineering courses currently in their evolution phases, it is an appropriate time to incorporate the teaching of GUI development skills into university-level education.

The benefits of introducing students to GUIs for educational purposes in engineering have been discussed by others [5], who note that GUIs enable an interactive visual approach to problem solving and thus can reinforce understanding of the governing theory. In the context of sports engineering, this idea should be taken one step further, and graduates should also be able to develop GUIs as a stand-alone learning outcome [6].

The motivation of this paper is to explain how the governing equations of motion for the trajectory of a lawn bowl can be used by students to present visual images of the bowl's trajectory through the development of a GUI. The aim of this work is to develop a learning tool for studying motion and developing GUI skills, with lawn bowls as the central theme.

2. Review of the Trajectory of a Lawn Bowl

The game of lawn bowls consists of rolling a biased bowl (*i.e.*, the bowl's centre of mass is not coincident with its geometric centre) along a grass surface. The trajectory of a lawn bowl poses an interesting problem that can be solved analytically. Following the methodology presented elsewhere [4], the key equations of motion may be summarised as;

$$v = v_o - \mu g t \tag{1}$$

$$T = v_o / \mu g \tag{2}$$

$$\phi = 2 \cdot p^{-1} \cdot \ln(v_0 / v) \tag{3}$$

$$p = 2 I_0 \mu / (MdR) \tag{4}$$

$$v_x = v \cdot \cos(\phi) \text{ and } v_y = v \cdot \sin(\phi) \tag{5}$$

where these variables are defined as in the nomenclature. From the numerical integration of equation 5 it is possible to establish the position of the bowl. Alternatively, by introducing the following equations the bowl position can be determined directly;

$$r_0 = v_0^2 p / (2\mu g) \tag{6}$$

$$x = r_0 \left(p - e^{-p\phi} \cdot p \cdot \cos\phi + e^{-p\phi} \cdot \sin\phi \right) / \left(1 + p^2 \right) \tag{7}$$

$$y = r_0 \left(1 - e^{-p\phi} \cdot p \cdot \sin\phi - e^{-p\phi} \cdot \cos\phi \right) / \left(1 + p^2 \right) \tag{8}$$

3. Implementation in MATLAB®

The implementation of the lawn bowl trajectory using a GUI in MATLAB® [7] is a multi-step process. The proposed methodology consists of the following phases. Depending on the skills and experience (and requirements) of the class, in may be appropriate to stop at a certain point, or skim-over some steps.

1) Implementation of equations; The derived equations must be entered into MATLAB (it is suggested using a script). This step helps students gain familiarity with MATLAB syntax. For beginners, this may require significant elucidation.
2) Graphical representation of trajectory; As well as simply generating a plot, in this step it is appropriate to demonstrate some of the formatting options available (colour, font size and type, line thickness, *etc.*).
3) Variation of parameters; Using the MATLAB script written in steps 1 & 2, various parameters should be experimented with so that students can develop an understanding of the trajectory sensitivity to these parameters. Students should be encouraged to describe the differences, and relate these to the physical effects.
4) Introduction to GUIs in MATLAB; The next step is to sequentially build the skills that are necessary to develop and operate a GUI in MATLAB. It is important to start with simple concepts and advance these progressively.
5) Implementation of GUI; Using the basic structure from step 4, the computational code from steps 1-3 should be incorporated. This can be done either as a separate function or embedded directly in the GUI code. In this step, it may be advantageous to allow multiple runs to be displayed to show the comparisons of different parameter changes.

As with any programming, there are multiple approaches that can be used to achieve the same end-result. Nonetheless, the approach suggested here should serve as an adequate guide. It is assumed that most instructors who would consider teaching this material would be themselves familiar with steps 1-3, and thus these are only given a brief mention.

3.1. Implementation of equations

For complete novices this is an appropriate introduction to defining variables in MATLAB. For more advanced students, this is also an opportunity to highlight that all constants should be declared together, and that these should be easy to locate in the code, likely near the beginning. For example, '`g=9.8; % gravity (m/s²)`' may be used to define a constant gravity of 9.8 m/s². Although this constant could easily be added into each equation within the code separately, by deliberating assigning a specific variable and giving it a meaningful name ensures it is clear what the number represents. The other advantage is that changing a constant at a later stage will require only changing a single number at the point at which it was declared, rather than hunting through the code and editing every instance.

Following the declaration of constants, variables may be progressively added to the code. Again, for novice students this is an opportunity to describe the differences between scalars, vectors and matrices. It is also appropriate to remind students who are familiar with MATLAB why pre-allocation of variables is advantageous.

Determining the bowl location by implementing both the numerical integration of equation 5 (with, for example, a `for` loop), and comparing with the algebraic approach (equations 7 & 8), by using the `tic/toc` commands it is appropriate to demonstrate to the students the advantages of minimising the use of "loops" in MATLAB, where possible. It is also possible to compare the relative accuracy of the two techniques, and the effect of the step-size used in the numerical integration.

3.2. Graphical representation of trajectory

Depending on the experience of the students, plotting the data from step 1 (implementation of equations) may be done by simply calling the plot command (at least as a starting point). This step also enables the introduction of the concepts of figures, axes, and incorporating the use of handles (including `get` and `set`). Additional features to improve the appearance of the figure should also be taught, including adding labels and units, tick locations, colours, linewidth, *etc*. Once familiar with the basic commands, students should be encouraged to expand upon this knowledge by further exploring figure properties independently and presenting graphical output in their own style.

An additional aspect of figures that many students may require some guidance with is the use of the `position` command, and how this differs for figures, axes, and other objects.

3.3. Variation of parameters

Changing various parameters (launch velocity, ball mass, *etc.*) in the MATLAB script should demonstrate the sensitivity of the bowl trajectory to these. Most of these parameter variations lead to intuitive results (*e.g.* increasing the speed increases the range) but a notable point to make is that if the axes are automatically scaled the resultant trajectory on the screen can be misleading. If not already covered in step 2, students should be introduced to the concept of `axis image`.

From a visualisation perspective, although *x*- and *y*-directions are defined in accordance with Cross [1], it may be better if the ball appeared to be launched vertically on the screen (as if the viewer was physically bowling it). Students should readily be able to make this coordinate frame change.

If it is not necessary to develop a GUI, this is an appropriate step at which to stop. For further work, the script developed hitherto could be converted to a function – with ball parameters as input variables. In either case, the developed *M*-file should be saved and kept separately before moving to the next task.

3.4. Introduction to GUIs in MATLAB

Implementation of GUIs in MATLAB can be achieved either entirely through scripting commands, or alternatively with the use of the GUI design environment (`GUIDE`). For educational purposes, it is most beneficial to teach students the fundamental skills of writing a GUI at command-level.

The most important thing to remember when teaching GUI development is to start simple. Before beginning to implement a GUI, the use of functions will be required. As such, it is important that students are familiar with writing functions before continuing. Using the `nargin` function will also prove beneficial. Knowledge of MATLAB 'structures' is also advantageous, but not essential.

Begin by creating very simple GUI objects, such as a textbox. An example of a basic introduction to GUI user-input, the following commands should be entered directly into the MATLAB command line;

```
guifig = figure;
struct.a = uicontrol('style','edit','String','3');
```

Students will soon learn that changing the value in the figure does not appear to have any effect. In order for the entered text to be read, get the value from the command line.

```
get(struct.a,'String')
```

Once learning how to extract information entered into a `uicontrol` object, the next step is to automate actions once a value has been modified – introducing students to the use of 'callbacks'. Again, start with a simple callback, for example;

```
struct.a=uicontrol('style','edit','String','3','callback','disp(''a'')');
```

Although not directly related to GUIs, it will also be appropriate for students to be aware of the '`userdata`' feature associated with figures, and how data may be attached to a figure using the `get` and

```
function step4(callstr)
if nargin==0
  guifig = figure;
  struct.a = uicontrol('style',...
  'edit','String','0',...
  'callback','step4(''foobar'')');
  set(guifig,'userdata',struct);
else
  struct = get(gcf,'userdata');
  get(struct.a,'String')
end
```

Fig. 1: Simple MATLAB function

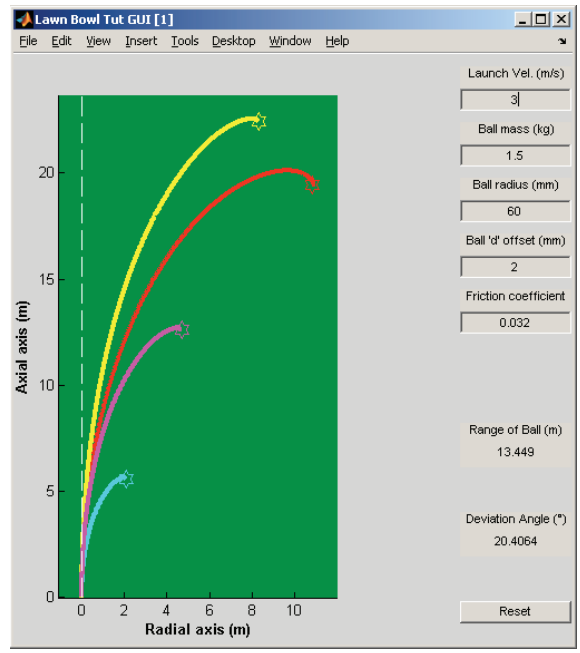

Fig. 2: Sample graphical user-interface for presenting lawn bowl trajectory (prepared in MATLAB)

`set` commands. This will prove useful for more advanced GUIs. The simple function shown in Fig. 1 demonstrates many of the features that will be a valuable basis for formulating a GUI. Note how this inherently is able to cope with multiple figures 'open'.

Before moving to step 5, it is advisable to first get students to write a simple GUI which adds two values. Making use of two text boxes (noting that the position of these will need to be set) add a '`text`' uicontrol to display the result of adding the two entered numbers. Remember that the entered value is a string, which will need to be converted to a number before addition. Attention is drawn to the MATLAB Central website [8], which is an excellent resource to obtain guidance in writing scripts and functions, including GUIs. The recommended simple GUI that adds two numbers, as mentioned above, is hosted on the MATLAB Central website, listed as 'APCST2011 GUI tutorial'.

Although not essential, students should note what happens when the figure is resized. Clearly this may have limitations for the user if the textboxes disappear from the figure. The figure properties allow the use of a '`resizefcn`' which can execute a callback when the figure size is changed. This can be used to callback another section of the code (for example through the use of a switch/case statement) in the created function that re-positions the uicontrol. It is left to instructors to guide students through the process of moving uicontrol objects when the figure size is changed, if deemed appropriate.

3.5. Implementation of a GUI

By combining the trajectory analysis introduced in steps 1-3 with the MATLAB GUI skills from step 4, students should be capable of creating a fully functioning GUI. There are many features and approaches that could be used. Students should be encouraged to use some initiative and creativity in this task. As always, students should start relatively simple, adding features one-by-one, checking functionality at each step.

An example of a GUI is shown in Figure 1. The preparsed pseudocode file is available from the MATLAB Central website [8], listed as 'Lawn Bowl Tutorial GUI'. In this sample, the launch velocity, ball parameters and the coefficient of friction may be varied. Multiple trajectories can be displayed simultaneously. The input parameters may be reset to default values. This is only provided as an example, and there are many variations that students should be encouraged to explore.

4. Summary

This paper has demonstrated the development of a graphical user interface (GUI) to present the trajectory of a lawn bowl. The motivation of this work is to incorporate skills in GUI development into undergraduate sports engineering programmes. Since sports engineering graduates will be required to work closely with athletes and coaches, it is imperative that they are able to quickly and efficiently process, analyse and display data, which can be effectively achieved through GUIs. The ability to develop GUIs is an area that may not be taught in existing engineering programmes, however it is essential that this ability is integrated into sports engineering programmes.

The outline to the GUI development presented in this paper has been performed in MATLAB®, although other computer programmes could equally be used to achieve the same outcome. Once armed with the fundamental skills introduced in this paper, students have the capabilities to extend their newly found skills to develop a range of other GUIs for a range of sports engineering applications.

Although this paper was written specifically to target tertiary-level engineering students, the final GUI in its fully developed form can be used as a stand-alone teaching aid for students without a programming background. The bias of the ball creates an interesting mathematical problem that is amenable to an analytical solution to the equations of motion. Aspects of the derivation of the lawn bowl trajectory, presented by Cross [4], are also appropriate for secondary school students in mathematics and physics. Particularly when accompanied by GUIs, examples involving sport in the areas of mathematics and physics can be included into secondary school education, and thus increase interest in the specific discipline of sports engineering, and engineering in general [9].

References

[1] Jenkins, P.E., Plaseided, A. and Khodaee, M. UCD Sports Engineering Program. *8th Conference of the International Sports Engineering Association (ISEA): Proceedia Engineering* 2010; **2**: 2757-2762.

[2] James, D.M. and Haake, S.J. Using Sport to Educate and Enthuse Young People About Engineering and the Physical Sciences. *The Engineering of Sport* 2006; **6**: 273-278.

[3] Rydakov, R., Nyashin, Y., Ilyalov, O. and Podgaets, R. Problems of Sport Engineering in Teaching Theoretical Mechanics. *8th Conference of the International Sports Engineering Association (ISEA): Proceedia Engineering* 2010; **2**: 2763-2768.

[4] Cross, R.The trajectory of a ball in lawn bowls. *American Journal of Physics* 1998; **66**(8):735-738.

[5] Azemi, A. and Yaz, E. E. Using Graphical User Interface Capabilities of MATLAB in Advanced Electrical Engineering Courses, *Proceedings of the 38th IEEE Conference on Decision and Control* 1999: 359-363.

[6] Smith, S. T. MATLAB: advanced GUI development, *Dog Ear Publishing*, LLC, 2006.

[7] MATLAB®, *The MathWorks Inc*.

[8] MATLAB Central, *The MathWorks Inc*, http://www.mathworks.com/matlabcentral/fileexchange/

[9] Cooke, A. and Taylor, A.M.B. Sports Engineering in Education: A Methodology for Educating Engineers and Promoting Engineering Careers. *IEE 2nd Annual Symposium on Engineering Education: Professional Engineering Scenarios* 2002; **1**: 12/1-12/7.